

AHG12: On uniform tile spacing

Yuji Fujimoto (Yuji.Fujimoto@sony.com)

Masaru Ikeda (Masaru.Ikeda@sony.com)

Teruhiko Suzuki (teruhiko.s@sony.com)

Introduction

- Background
 - In the HEVC specification, when uniform tile spacing is set equal to 1 (`uniform_spacing_flag = 1`), The number of tile rows and columns are signaled in PPS. A picture is partitioned into tiles will almost equal number of CTUs.
 - In the current VVC draft5 (JVET-N1001), when uniform tile spacing is set equal to 1 (`uniform_tile_split_flag = 1`), tile width of columns and height of rows are signaled in PPS.
- Problems
 - In HEVC, The tile boundary position could be changed from the original position.
 - In case of VVC, The above problem is resolved as proposed by JVET-N0498 proposal. But when the picture size is not a multiple of indicated tile size, right column and bottom row tiles could be much smaller than other tiles. Such not-well balanced tile partition is not good for parallel processing and uniform bit allocation use case.

Uniform tile spacing in HEVC

```

pic_parameter_set_rbsp() {
    . . . .
    tiles_enabled_flag
    entropy_coding_sync_enabled_flag
    if( tiles_enabled_flag ) {
        num_tile_columns_minus1
        num_tile_rows_minus1
        uniform_spacing_flag
        if( !uniform_spacing_flag ) {
            for( i = 0; i < num_tile_columns_minus1; i++ )
                column_width_minus1[ i ]
            for( i = 0; i < num_tile_rows_minus1; i++ )
                row_height_minus1[ i ]
        }
    }
    . . . .

```

Tile partition example :

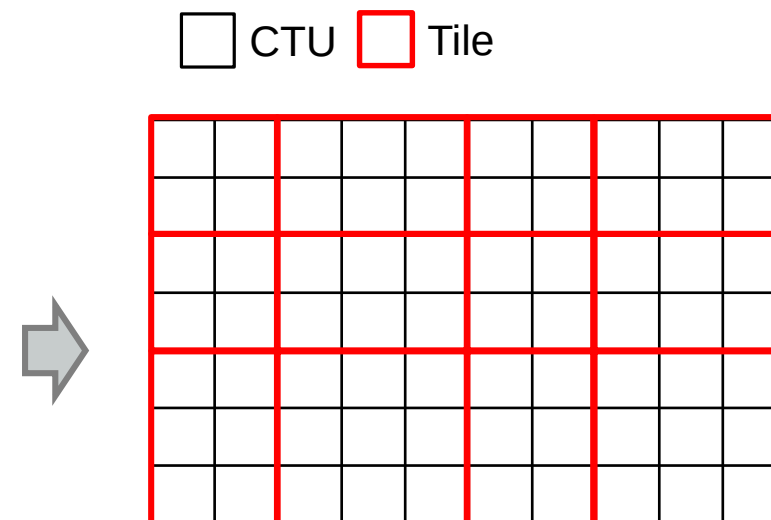
PicWidthInCtbsY = 10

PicHeightInCtbsY = 7

num_tile_columns_minus1 = 3

num_tile_rows_minus1 = 2

uniform_spacing_flag = 1



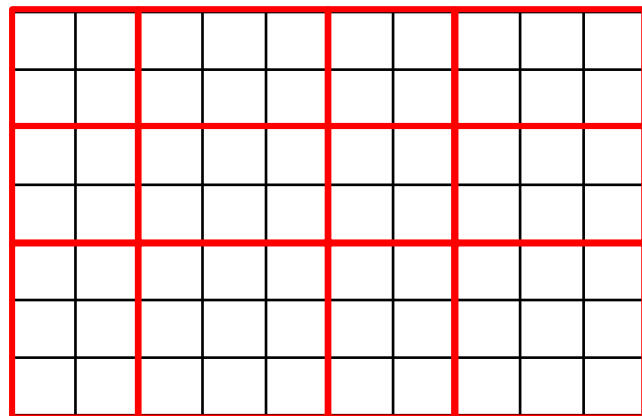
Tile partitioning method of uniform spacing in HEVC

```

if( uniform_tile_spacing_flag )
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        colWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 ) -
                        ( i * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 )
if( uniform_tile_spacing_flag )
    for( j = 0; j <= num_tile_rows_minus1; j++ )
        rowHeight[ j ] = ( ( j + 1 ) * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 ) -
                        ( j * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 )

```

The problem of uniform tile spacing in HEVC



□ CTU □ Tile □ Removed parts

Tile partition example :

PicWidthInCtbsY = 10

PicHeightInCtbsY = 7

num_tile_columns_minus1 = 3

num_tile_rows_minus1 = 2

uniform_spacing_flag = 1

Column width is derived according to specified tile partitioning method (p3) as follows

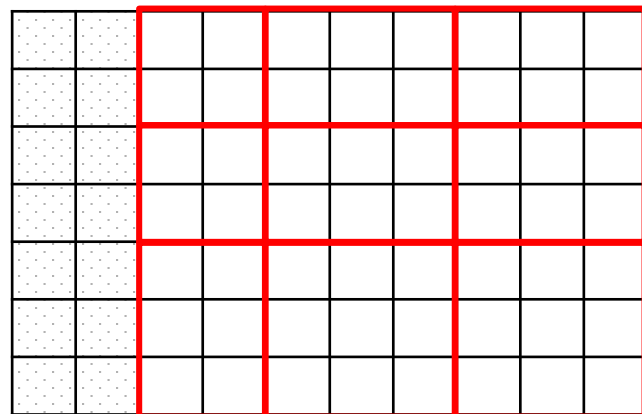
$$\text{colWidth}[0] = (1 \cdot 10) / 4 - (0 \cdot 10) / 4 = 2$$

$$\text{colWidth}[1] = (2 \cdot 10) / 4 - (1 \cdot 10) / 4 = 3$$

$$\text{colWidth}[2] = (3 \cdot 10) / 4 - (2 \cdot 10) / 4 = 2$$

$$\text{colWidth}[3] = (4 \cdot 10) / 4 - (3 \cdot 10) / 4 = 3$$

In the case of right side tile extraction(left tile column is discarded)



Tile partition example :

PicWidthInCtbsY = 8

PicHeightInCtbsY = 7

num_tile_columns_minus1 = 2

num_tile_rows_minus1 = 2

uniform_spacing_flag = 1

Column width is derived according to specified tile partitioning method (p3) as follows

$$\text{colWidth}[0] = (1 \cdot 8) / 3 - (0 \cdot 8) / 3 = 2$$

$$\text{colWidth}[1] = (2 \cdot 8) / 3 - (1 \cdot 8) / 3 = 3$$

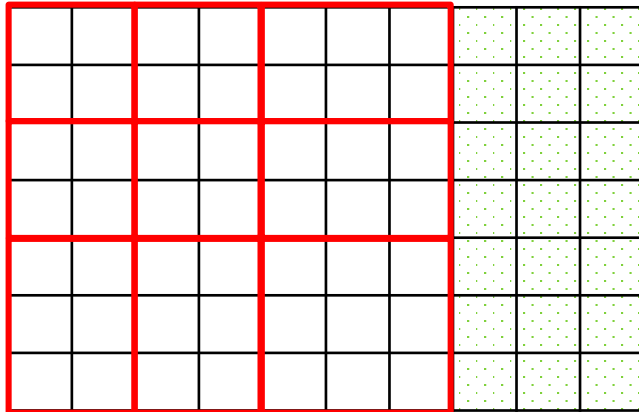
$$\text{colWidth}[2] = (3 \cdot 8) / 3 - (2 \cdot 8) / 3 = 3$$

Tile boundary is changed from the original position

The problem of uniform tile spacing in HEVC (Continued)

□ CTU □ Tile ■ Removed parts

In the case of left side tile extraction(right tile column is discarded)



Tile partition example :

$\text{PicWidthInCtbsY} = 7$

$\text{PicHeightInCtbsY} = 7$

$\text{num_tile_columns_minus1} = 2$

$\text{num_tile_rows_minus1} = 2$

$\text{uniform_spacing_flag} = 1$

Column width is derived according to specified tile partitioning method (p3) as follows

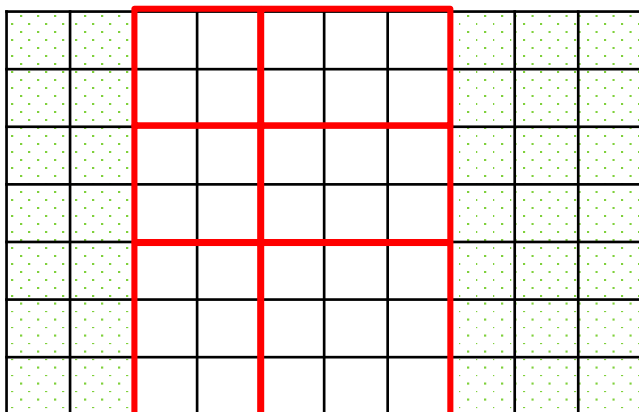
$$\text{colWidth}[0] = (1*7)/3 - (0*7)/3 = 2$$

$$\text{colWidth}[1] = (2*7)/3 - (1*7)/3 = 2$$

$$\text{colWidth}[2] = (3*7)/3 - (2*7)/3 = 3$$

Tile boundary is changed from the original position

In the case of middle tile extraction(both left and right columns are discarded)



Tile partition example :

$\text{PicWidthInCtbsY} = 5$

$\text{PicHeightInCtbsY} = 5$

$\text{num_tile_columns_minus1} = 1$

$\text{num_tile_rows_minus1} = 2$

$\text{uniform_spacing_flag} = 1$

Column width is derived according to specified tile partitioning method (p3) as follows

$$\text{colWidth}[0] = (1*5)/2 - (0*5)/2 = 2$$

$$\text{colWidth}[1] = (2*5)/2 - (1*5)/2 = 3$$

Tile boundary is changed from the original position

uniform tile spacing in current VVC draft5 (JVET-N1001)

```
pic_parameter_set_rbsp() {  
    . . . .  
    single_tile_in_pic_flag  
    if( !single_tile_in_pic_flag ) {  
        uniform_tile_spacing_flag  
        if( uniform_tile_spacing_flag ) {  
            tile_cols_width_minus1  
            tile_rows_height_minus1  
        } else {  
            num_tile_columns_minus1  
            num_tile_rows_minus1  
            for( i = 0; i < num_tile_columns_minus1; i++ )  
                tile_column_width_minus1[ i ]  
            for( i = 0; i < num_tile_rows_minus1; i++ )  
                tile_row_height_minus1[ i ]  
            . . . .  
        }  
    }  
}
```

Tile partition example :

PicWidthInCtbsY = 10

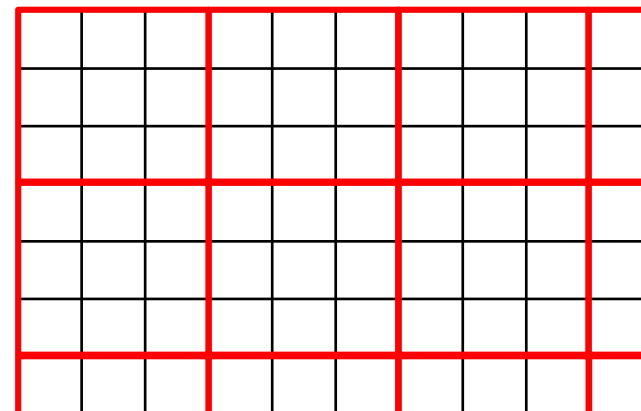
PicHeightInCtbsY = 7

uniform_tile_spacing_flag = 1

tile_cols_width_minus1 = 2

tile_rows_height_minus1 = 2

□ CTU □ Tile



Not-well balanced tile partition
⇒ Computational cost is not equally balanced

Tile partitioning method of uniform spacing in current VVC draft5

```
if( uniform_tile_spacing_flag ) {  
    remainingWidthInCtbsY = PicWidthInCtbsY  
    i = 0  
    while( remainingWidthInCtbsY > ( tile_cols_width_minus1 + 1 ) ) {  
        colWidth[ i++ ] = tile_cols_width_minus1 + 1  
        remainingWidthInCtbsY -= ( tile_cols_width_minus1 + 1 )  
    }  
    colWidth[ i ] = remainingWidthInCtbsY  
    num_tile_columns_minus1 = i  
}
```

```
if( uniform_tile_spacing_flag ) {  
    remainingHeightInCtbsY = PicHeightInCtbsY  
    i = 0  
    while( remainingHeightInCtbsY > ( tile_rows_height_minus1 + 1 ) ) {  
        rowHeight[ i++ ] = tile_rows_height_minus1 + 1  
        remainingHeightInCtbsY -= ( tile_rows_height_minus1 + 1 )  
    }  
    rowHeight[ i ] = remainingHeightInCtbsY  
    num_tile_rows_minus1 = i  
}
```

Proposed method

- Improved tile partitioning methods with uniform tile spacing for equally balanced parallel processing and uniform bit allocation
 - Tile boundary is not changed by extraction of tiles (or discarding tiles)
 - Tile extraction (or discarding tiles) can be done with few changes in PPS and can be done by simply discarding VCL NAL unit for tiles
- Option1 : Improved tile partitioning method with uniform tile spacing
- Option2 : Add information to support tile extraction without changing tile boundary. When bitstream does not support tile extraction without changing tile boundary, tile partitioning method is the same as HEVC. When it is supported, the additional information for original tile boundary is signaled in PPS
- Those methods can be applied to Brick partitioning with uniform spacing

Option1

- The improved tile partitioning method for uniform tile spacing. There could be three variants to derive tile column and row height.
 - The initial values of parameters are set at first while processing tile partition for the Picture size in CTU unit(PicWidthInCtbsY, PicHeightInCtbsY) with the specified number of tile column and row(tileColsWidth, tileRowsHeight) as follows.
remainWidthInCtbsY = PicWidthInCtbsY, remainHeightInCtbs = PicHeightInCtbsY
remainTileColumns = tileColsWidth, RemaingTileRows = tileRowsHeight
 - Option1-a width and height are **calculated by truncation** with left to right and up to down order
 - Each tile size(colWidth[i], rowHeight[j]) is derived repeatedly with each i,j as follows
colWidth[i] = remainWidthInCtbs/remaingTileColumns (truncated integer)
remainWidthInCtbsY -= colWidth[i], remaingTileColumns--
rowHeight[j] = remainHeightInCtbs/remaingTileRows (truncated integer)
remaingHeightInCtbsY -= rowHeight[j], remaingTileRows--
 - Option1-b width and height are **calculated by rounding up** with left to right and up to down order
 - Each tile size(colWidth[i], rowHeight[j]) is derived repeatedly with each i,j as follows
colWidth[i] = ceil(remainWidthInCtbs/remaingTileColumns) (round up)
remainWidthInCtbs -= colWidth[i], remaingTileColumns--
rowHeight[j] = ceil(remainHeightInCtbs/remaingTileRows) (round up)
remaingHeightInCtbsY -= rowHeight[j], remaingTileRows--
 - Option1-c
 - It's another variant. In this case, the flag in PPS to indicate which Options (1-a or 1-b) is used to derive tile width and height

Improved tile partitioning method (Option1-a)

PPS syntax is shown as below

pic_parameter_set_rbsp() {
. . .
single_tile_in_pic_flag
if(!single_tile_in_pic_flag) {
num_tile_columns_minus1
num_tile_rows_minus1
uniform_tile_spacing_flag
if(!uniform_tile_spacing_flag) {
for(i = 0; i < num_tile_columns_minus1; i++)
tile_column_width_minus1[i]
for(i = 0; i < num_tile_rows_minus1; i++)
tile_row_height_minus1[i]
. . .
}

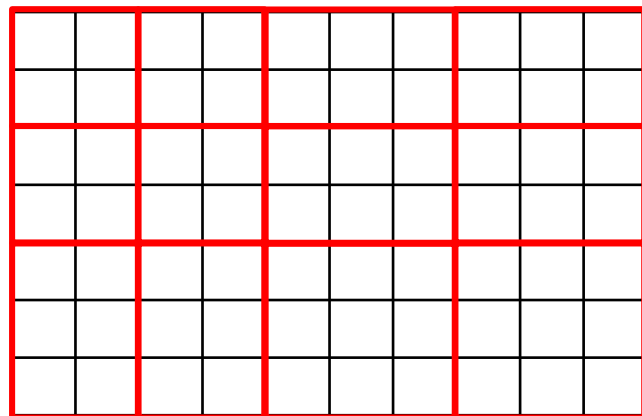
This is the same as HEVC syntax

Draft Text (6-1),(6-2) modification :
Column width row height are derived as follows
(width and height are calculated by truncation repeatedly)

```
if( uniform_tile_spacing_flag ) {
    remainingWidthInCtbsY = PicWidthInCtbsY
    remainingTileColumns = tile_cols_width_minus1 + 1
    i = 0
    while( remainingTileColumns > 0 ) {
        colWidth[ i ] = remainingWidthInCtbsY / remainingTileColumns
        remainingWidthInCtbsY -= colWidth[ i ]
        remainingTileColumns--
        i++
    }
}
```

```
if( uniform_tile_spacing_flag ) {
    remainingHeightInCtbsY = PicHeightInCtbsY
    remainingTileRows = tile_rows_height_minus1 + 1
    i = 0
    while( remainingTileRows > 0 ) {
        rowHeight[ i ] = remainingHeightInCtbsY / remainingTileRows
        remainingHeightInCtbsY -= rowHeight[ i ]
        remainingTileRows--
        i++
    }
}
```

Tile partitioning example of Option1-a



□ CTU □ Tile ■ Removed parts

Tile partition example :

PicWidthInCtbsY = 10

PicHeightInCtbsY = 7

num_tile_columns_minus1 = 3

num_tile_rows_minus1 = 2

uniform_tile_spacing_flag = 1

Column width is derived according to Option1-a tile partitioning method (p9) as follows

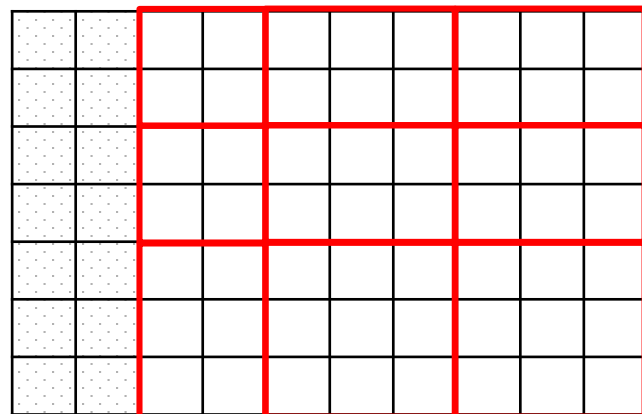
$\text{colWidth}[0] = 10/4 = 2$

$\text{colWidth}[1] = 8/3 = 2$

$\text{colWidth}[2] = 6/2 = 3$

$\text{colWidth}[3] = 3/1 = 3$

In the case of right side tile extraction(left tile column is discarded)



Tile partition example :

PicWidthInCtbsY = 8

PicHeightInCtbsY = 7

num_tile_columns_minus1 = 2

num_tile_rows_minus1 = 2

uniform_tile_spacing_flag = 1

Column width is derived according to Option1-a tile partitioning method (p9) as follows

$\text{colWidth}[0] = 8/3 = 2$

$\text{colWidth}[1] = 6/3 = 3$

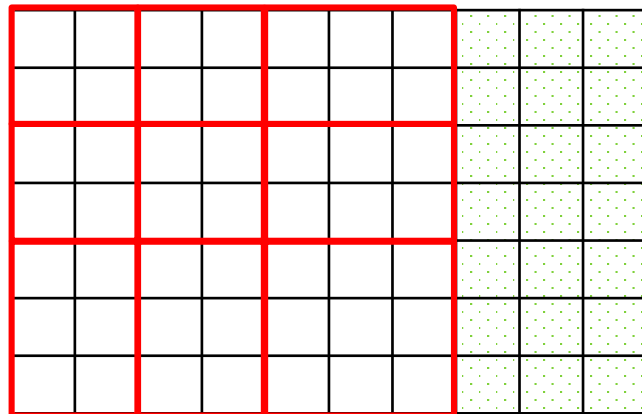
$\text{colWidth}[2] = 3/1 = 3$

Tile boundary is not changed from the original position

Tile partitioning example of Option1-a (Continued)

□ CTU □ Tile □ Removed parts

In the case of left side tile extraction(right tile column is discarded)



Tile partition example :

PicWidthInCtbsY = 7
PicHeightInCtbsY = 7
num_tile_columns_minus1 = 2
num_tile_rows_minus1 = 2
uniform_tile_spacing_flag = 1

Column width is derived according to Option1-a tile partitioning method (p9) as follows

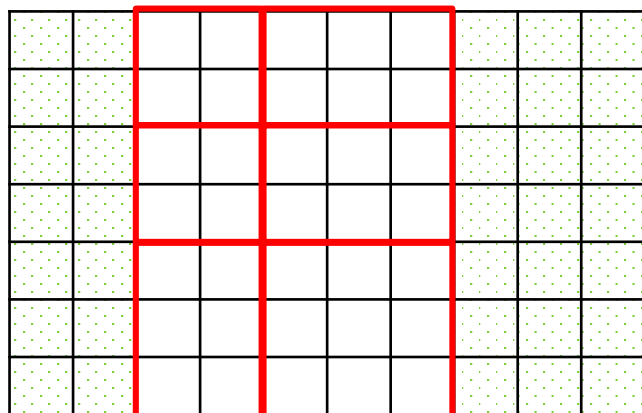
$$\text{colWidth}[0] = 7/3 = 2$$

$$\text{colWidth}[1] = 5/2 = 2$$

$$\text{colWidth}[2] = 3/1 = 3$$

Tile boundary is not changed from the original position

In the case of middle tile extraction(both left and right columns are discarded)



Tile partition example :

PicWidthInCtbsY = 5
PicHeightInCtbsY = 7
num_tile_columns_minus1 = 1
num_tile_rows_minus1 = 2
uniform_tile_spacing_flag = 1

Column width is derived according to Option1-a tile partitioning method (p9) as follows

$$\text{colWidth}[0] = 5/2 = 2$$

$$\text{colWidth}[1] = 3/1 = 3$$

Tile boundary is not changed from the original position

Improved tile partitioning method (Option1-b)

PPS syntax is shown as below

pic_parameter_set_rbsp() {
. . .
single_tile_in_pic_flag
if(!single_tile_in_pic_flag) {
num_tile_columns_minus1
num_tile_rows_minus1
uniform_tile_spacing_flag
if(!uniform_tile_spacing_flag) {
for(i = 0; i < num_tile_columns_minus1; i++)
tile_column_width_minus1[i]
for(i = 0; i < num_tile_rows_minus1; i++)
tile_row_height_minus1[i]
. . .
}

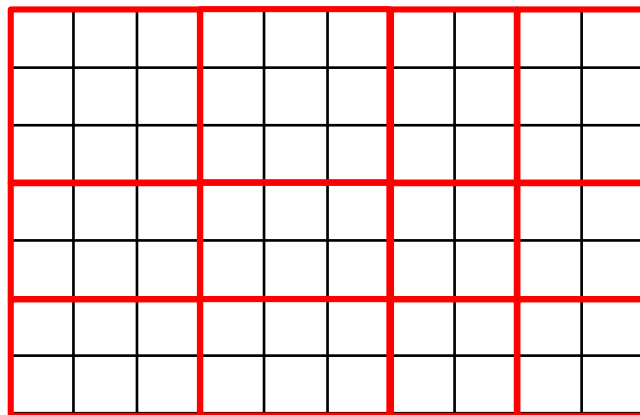
This is the same as HEVC syntax

Draft Text (6-1),(6-2) modification :
Column width row height are derived as follows
(width and height are calculated by round up repeatedly)

```
if( uniform_tile_spacing_flag ) {
    remainingWidthInCtbsY = PicWidthInCtbsY
    remainingTileColumns = tile_cols_width_minus1 + 1
    i = 0
    while( remainingTileColumns > 0 ) {
        colWidth[ i ] = ceil(remainingWidthInCtbsY/ remainingTileColumns)
        remainingWidthInCtbsY -= colWidth[ i ]
        remainingTileColumns--
        i++
    }
}
```

```
if( uniform_tile_spacing_flag ) {
    remainingHeightInCtbsY = PicHeightInCtbsY
    remainingTileRows = tile_rows_height_minus1 + 1
    i = 0
    while( remainingTileRows > 0 ) {
        rowHeight[ i ] = ceil(remainingHeightInCtbsY/ remainingTileRows)
        remainingHeightInCtbsY -= rowHeight[ i ]
        remainingTileRows--
        i++
    }
}
```

Tile partitioning example of Option1-b



□ CTU □ Tile □ Removed parts

Tile partition example :

PicWidthInCtbsY = 10

PicHeightInCtbsY = 7

```
num_tile_columns_minus1 = 3
```

```
num_tile rows minus1 = 2
```

uniform tile spacing flag = 1

Column width is derived according to Option1-b tile partitioning method (p12) as follows

```
colWidth[ 0 ] = ceil(10/4) = 3
```

```
colWidth[ 1 ] = ceil(7/3) = 3
```

$$\text{colWidth}[2] = \text{ceil}(4/2) = 2$$

```
colWidth[ 3 ] = ceil(2/1) = 2
```

In the case of right side tile extraction(left tile column is discarded)



Tile partition example :

PicWidthInCtbsY = 7

PicHeightInCtbsY = 7

```
num_tile_columns_minus1 = 2
```

```
num_tile_rows_minus1 = 2
```

uniform_tile_spacing_flag = 1

Column width is derived according to Option1-b tile partitioning method (p12) as follows

```
colWidth[ 0 ] = ceil(7/3) = 3
```

```
colWidth[ 1 ] = ceil(4/2) = 2
```

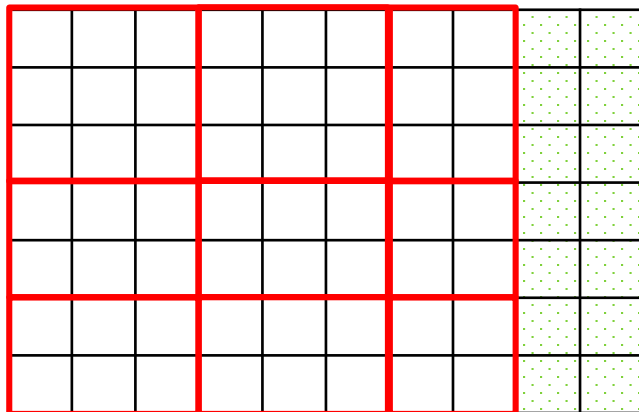
```
colWidth[ 2 ] = ceil(2/1) = 2
```

Tile boundary is not changed from the original position

Tile partitioning example of Option1-b (Continued)

□ CTU □ Tile □ Removed parts

In the case of left side tile extraction(right tile column is discarded)



Tile partition example :

PicWidthInCtbsY = 8
PicHeightInCtbsY = 7
num_tile_columns_minus1 = 2
num_tile_rows_minus1 = 2
uniform_tile_spacing_flag = 1

Column width is derived according to Option1-b tile partitioning method (p12) as follows

colWidth[0] = $\text{ceil}(8/3) = 3$
colWidth[1] = $\text{ceil}(5/2) = 3$
colWidth[2] = $\text{ceil}(3/1) = 3$

Tile boundary is not changed from the original position

In the case of middle tile extraction(both left and right columns are discarded)



Tile partition example :

PicWidthInCtbsY = 5
PicHeightInCtbsY = 7
num_tile_columns_minus1 = 1
num_tile_rows_minus1 = 2
uniform_tile_spacing_flag = 1

Column width is derived according to Option1-b tile partitioning method (p12) as follows

colWidth[0] = $\text{ceil}(5/2) = 3$
colWidth[1] = $\text{ceil}(3/1) = 3$

Tile boundary is not changed from the original position

Option1-c: Option1-a or Option1-b indication

PPS syntax is shown as below

pic_parameter_set_rbsp() {	
. . .	
single_tile_in_pic_flag	u(1)
if(!single_tile_in_pic_flag) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_tile_spacing_flag	u(1)
if(uniform_tile_spacing_flag) {	
uniform_tile_spacing_mode	u(1)
} else {	
for(i = 0; i < num_tile_columns_minus1; i++)	
tile_column_width_minus1[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
tile_row_height_minus1[i]	ue(v)
. . .	

uniform_tile_spacing_mode :
A flag to indicate which Options (1-a or 1-b)
is used

Option2

- Add information to support tile extraction without changing tile boundary
 - Add Flags to indentify if bitstreams supports tile extraction
 - When bitstream does not support tile, tile partitioning method is the same as HEVC
 - When HEVC bitstream is transcoded to VVC bitstream, it is not necessary to change its tile partitioning
- According to added information, tile boundary is not changed by extraction of tiles
 - It requires more changes in PPS compared to Option1 for tile extraction

Option2: Add information to support tile extraction without changing tile boundary

PPS syntax is shown as below

pic_parameter_set_rbsp() {	
. . . .	
single_tile_in_pic_flag	u(1)
if(!single_tile_in_pic_flag) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_tile_spacing_flag	u(1)
if(uniform_tile_spacing_flag) {	
uniform_tile_spacing_extract_column_flag	u(1)
if(uniform_tile_spacing_extract_column_flag) {	
uniform_tile_spacing_extract_column_start_offset	ue(v)
uniform_tile_spacing_extract_width_in_ctb	ue(v)
uniform_tile_spacing_extract_tile_column_offset	ue(v)
}	
uniform_tile_spacing_extract_row_flag	u(1)
if(uniform_tile_spacing_extract_row_flag) {	
uniform_tile_spacing_extract_row_start_offset	ue(v)
uniform_tile_spacing_extract_height_in_ctb	ue(v)
uniform_tile_spacing_extract_tile_row_offset	ue(v)
}	
} else {	
for(i = 0; i < num_tile_columns_minus1; i++)	
tile_column_width_minus1[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
tile_row_height_minus1[i]	ue(v)
. . . .	

Draft Text (6-1),(6-2) modification :

Column width row height are derived as follows

(When bitstream does not support tile extraction, tile partitioning method is the same as HEVC)

```

if( uniform_tile_spacing_flag ){
    PicWidthInCtbsY2 = PicWidthInCtbsY + uniform_tile_spacing_extract_width_in_ctb
    numTileColumns2 = num_tile_columns_minus1 + 1 + uniform_tile_spacing_extract_tile_column_offset
    for( i = 0; i <= num_tile_columns_minus1; i++ ){
        i2 = i + uniform_tile_spacing_extract_column_start_offset
        ColWidth[ i ] = ( ( i2 + 1 ) * PicWidthInCtbsY2 ) / numTileColumns2 -
                        ( i2 * PicWidthInCtbsY2 ) / numTileColumns2
    }
}

```

```

if( uniform_tile_spacing_flag ){
    PicHeightInCtbsY2 = PicHeightInCtbsY + uniform_tile_spacing_extract_height_in_ctb
    numTileRows2 = num_tile_rows_minus1 + 1 + uniform_tile_spacing_extract_tile_row_offset
    for( j = 0; j <= num_tile_rows_minus1; j++ ){
        j2 = j + uniform_tile_spacing_extract_row_start_offset
        RowHeight[ j ] = ( ( j2 + 1 ) * PicHeightInCtbsY2 ) / numTileRows2 -
                        ( j2 * PicHeightInCtbsY2 ) / numTileRows2
    }
}

```

```

if( !uniform_tile_spacing_extract_column_flag ){
    uniform_tile_spacing_extract_column_start_offset = 0
    uniform_tile_spacing_extract_width_in_ctb = 0
    uniform_tile_spacing_extract_tile_column_offset = 0
}

```

```

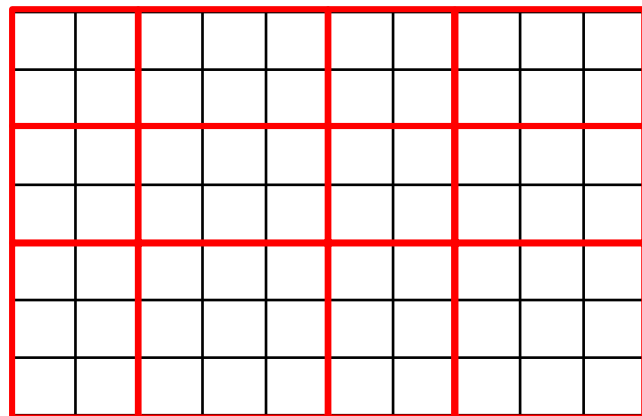
if( !uniform_tile_spacing_extract_row_flag ){
    uniform_tile_spacing_extract_row_start_offset = 0
    uniform_tile_spacing_extract_height_in_ctb = 0
    uniform_tile_spacing_extract_tile_row_offset = 0
}

```

Additional information in Option2

- The semantics of the additional syntax elements are as follows
 - `uniform_tile_spacing_extract_column_flag` : A flag to identify if bitstream supports tile extraction without changing vertical tile boundary
 - `uniform_tile_spacing_extract_column_start_offset` : The location that can be extracted tile column
 - `uniform_tile_spacing_extract_width_in_ctb` : The number of removing CTB columns of tile column
 - `uniform_tile_spacing_extract_tile_column_offset` : The number of removing tile columns of tile column extraction
 - `uniform_tile_spacing_extract_row_flag` : A flag to identify if bitstream supports tile extraction without changing horizontal tile boundary
 - `uniform_tile_spacing_extract_row_start_offset` : The location that can be extracted tile row
 - `uniform_tile_spacing_extract_height_in_ctb` : The number of removing CTB rows of tile row extraction
 - `uniform_tile_spacing_extract_tile_row_offset` : The number of removing tile rows of tile row extraction

Tile partitioning example of Option2



□ CTU □ Tile □ Removed parts

Tile partition example :

PicWidthInCtbsY = 10

PicHeightInCtbsY = 7

num_tile_columns_minus1 = 3

num_tile_rows_minus1 = 2

uniform_tile_spacing_flag = 1

uniform_tile_spacing_extract_column_flag = 0

uniform_tile_spacing_extract_row_flag = 0

Column width is derived according to Option2 tile partitioning method (p17) as follows

$$\text{colWidth}[0] = (1 \cdot 10) / 4 - (0 \cdot 10) / 4 = 2$$

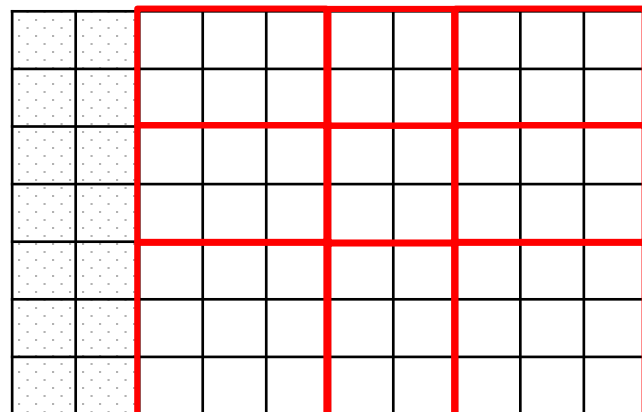
$$\text{colWidth}[1] = (2 \cdot 10) / 4 - (1 \cdot 10) / 4 = 3$$

$$\text{colWidth}[2] = (3 \cdot 10) / 4 - (2 \cdot 10) / 4 = 2$$

$$\text{colWidth}[3] = (4 \cdot 10) / 4 - (3 \cdot 10) / 4 = 3$$

Tile boundary is the same as HEVC

In the case of right side tile extraction(left tile column is discarded)



Tile partition example :

PicWidthInCtbsY = 8

PicHeightInCtbsY = 7

num_tile_columns_minus1 = 2

num_tile_rows_minus1 = 2

uniform_tile_spacing_flag = 1

uniform_tile_spacing_extract_column_flag = 1

uniform_tile_spacing_extract_column_start_offset = 1

uniform_tile_spacing_extract_width_in_ctb = 2

uniform_tile_spacing_extract_tile_column_offset = 1

uniform_tile_spacing_extract_row_flag = 0

Column width is derived according to Option2 tile partitioning method (p17) as follows

$$\text{colWidth}[0] = ((1+1) \cdot (8+2)) / (3+1) - ((0+1) \cdot (8+2)) / (3+1) = 3$$

$$\text{colWidth}[1] = ((2+1) \cdot (8+2)) / (3+1) - ((1+1) \cdot (8+2)) / (3+1) = 2$$

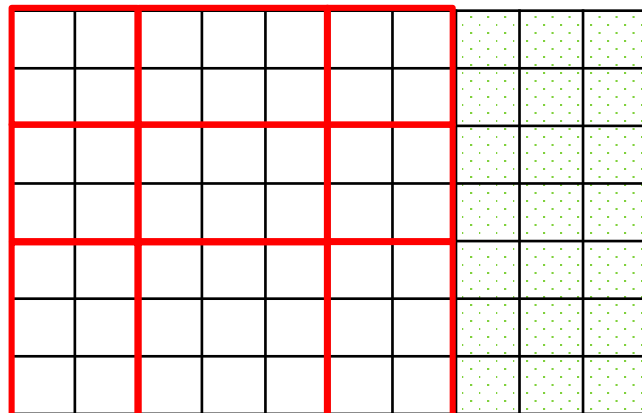
$$\text{colWidth}[2] = ((3+1) \cdot (8+2)) / (3+1) - ((2+1) \cdot (8+2)) / (3+1) = 3$$

Tile boundary is not changed from the original position

Tile partitioning example of Option2 (Continued)

□ CTU □ Tile ■ Removed parts

In the case of left side tile extraction(right tile column is discarded)



Tile partition example :

PicWidthInCtbsY = 7
PicHeightInCtbsY = 7
num_tile_columns_minus1 = 2
num_tile_rows_minus1 = 2
uniform_tile_spacing_flag = 1
uniform_tile_spacing_extract_column_flag = 1
uniform_tile_spacing_extract_column_start_offset = 0
uniform_tile_spacing_extract_width_in_ctb = 3
uniform_tile_spacing_extract_tile_column_offset = 1
uniform_tile_spacing_extract_row_flag = 0

Column width is derived according to Option2 tile partitioning method (p17) as follows

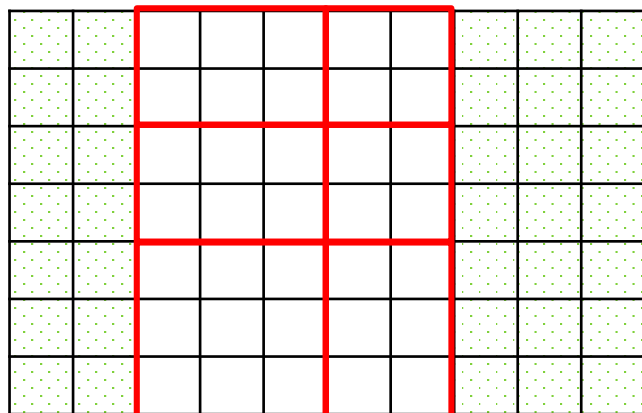
$$\text{colWidth}[0] = (1*(7+3))/(3+1) - (0*(7+3))/(3+1) = 2$$

$$\text{colWidth}[1] = (2*(7+3))/(3+1) - (1*(7+3))/(3+1) = 3$$

$$\text{colWidth}[2] = (3*(7+3))/(3+1) - (2*(7+3))/(3+1) = 2$$

Tile boundary is not changed from the original position

In the case of middle tile extraction(both left and right columns are discarded)



Tile partition example :

PicWidthInCtbsY = 5
PicHeightInCtbsY = 7
num_tile_columns_minus1 = 1
num_tile_rows_minus1 = 2
uniform_tile_spacing_flag = 1
uniform_tile_spacing_extract_column_flag = 1
uniform_tile_spacing_extract_column_start_offset = 1
uniform_tile_spacing_extract_width_in_ctb = 5
uniform_tile_spacing_extract_tile_column_offset = 2
uniform_tile_spacing_extract_row_flag = 0

Column width is derived according to Option2 tile partitioning method (p17) as follows

$$\text{colWidth}[0] = ((1+1)*(5+5))/(2+2) - ((0+1)*(5+5))/(2+2) = 3$$

$$\text{colWidth}[1] = ((2+1)*(5+5))/(2+2) - ((1+1)*(5+5))/(2+2) = 2$$

Tile boundary is not changed from the original position

Uniform spacing for brick

PPS syntax for Option1-a and Option1-b is as follows

pic_parameter_set_rbsp() {
. . . .
brick_splitting_present_flag
for(i = 0; brick_splitting_present_flag && i < NumTilesInPic; i++) {
brick_split_flag[i]
if(brick_split_flag[i]) {
num_brick_rows_minus1[i]
uniform_brick_spacing_flag[i]
if(!uniform_brick_spacing_flag[i]){
for(j = 0; j < num_brick_rows_minus1[i]; j++)
brick_row_height_minus1[i][j]
}
}
}
}

Draft Text (6-5) modification for Option1-a

```

if( uniform_brick_spacing_flag[ i ] ) {
    remainingHeightInCtbsY = RowHeight[ tileY ]
    remainingTileRows = num_brick_rows_minus1[i] + 1
    j = 0
    while( remainingTileRows > 0 ) {
        rowHeight2[ j ] = remainingHeightInCtbsY / remainingTileRows
        remainingHeightInCtbsY -= rowHeight2[ j ]
        remainingTileRows--
        j++
    }
}

```

PPS syntax for Option1-c

pic_parameter_set_rbsp() {	
. . . .	
brick_splitting_present_flag	u(1)
for(i = 0; brick_splitting_present_flag && i < NumTilesInPic; i++) {	
brick_split_flag[i]	u(1)
if(brick_split_flag[i]) {	
num_brick_rows_minus1[i]	ue(v)
uniform_brick_spacing_flag[i]	u(1)
if(uniform_brick_spacing_flag[i]){	
uniform_brick_spacing_mode	u(1)
} else {	
for(j = 0; j < num_brick_rows_minus1[i]; j++)	
brick_row_height_minus1[i][j]	ue(v)
}	
}	
}	
}	

Draft Text (6-5) modification for Option1-b

```

if( uniform_brick_spacing_flag[ i ] ) {
    remainingHeightInCtbsY = RowHeight[ tileY ]
    remainingTileRows = num_brick_rows_minus1[i] + 1
    j = 0
    while( remainingTileRows > 0 ) {
        rowHeight2[ j ] = ceil(remainingHeightInCtbsY / remainingTileRows)
        remainingHeightInCtbsY -= rowHeight2[ j ]
        remainingTileRows--
        j++
    }
}

```

Uniform spacing for brick (Continued)

PPS syntax for Option2

pic_parameter_set_rbsp() {	
. . .	
brick_splitting_present_flag	u(1)
for(i = 0; brick_splitting_present_flag && i < NumTilesInPic; i++)	
{	
brick_split_flag[i]	u(1)
if(brick_split_flag[i]) {	
num_brick_rows_minus1[i]	ue(v)
uniform_brick_spacing_flag[i]	u(1)
if(uniform_brick_spacing_flag[i]) {	
uniform_brick_spacing_extract_flag[i]	u(1)
if(uniform_brick_spacing_extract_flag[i]) {	
uniform_brick_spacing_extract_row_start_offset[i]	ue(v)
uniform_brick_spacing_extract_height_in_ctb[i]	ue(v)
uniform_brick_spacing_extract_brick_row_offset[i]	ue(v)
}	
} else {	
for(j = 0; j < num_brick_rows_minus1[i]; j++)	
brick_row_height_minus1[i][j]	ue(v)
}	
}	
}	

Draft Text (6-5) modification for Option2

```

if( uniform_brick_spacing_flag[ i ] ){
    RowHeight2 = RowHeight[ tileY ] + uniform_brick_spacing_extract_height_in_ctb[ i ]
    numBrickRows2 = num_brick_rows_minus1[ i ] + 1 + uniform_brick_spacing_extract_brick_row_offset[ i ]
    for( j = 0; j <= num_brick_rows_minus1[ i ]; j++ ){
        j2 = j + uniform_brick_spacing_extract_row_start_offset[ i ]
        rowHeight2[ j ] = ( ( j2 + 1 ) * RowHeight2 ) / numBrickRows2 -
            ( j2 * RowHeight2 ) / numBrickRows2
    }
}

if ( !uniform_brick_spacing_extract_flag[ i ] ){
    uniform_brick_spacing_extract_row_start_offset[ i ] = 0
    uniform_brick_spacing_extract_height_in_ctb[ i ] = 0
    uniform_brick_spacing_extract_tile_row_offset[ i ] = 0
}

```